This tutorial is supposed to help you learning to use the framework AngularJS in your projects. You can find a step by step guide on how to use the framework.

Step by step guide

- **Step 1: Integration of the Framework**
  You can easily add the library of the AngularJS framework to your project. The only thing that you have to do is to embed a script in the head-section of your HTML-file. Since there is an online source of the library, you don't need to have the local file.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="ISO-8859-1">
5     <title>Example AngularJS</title>
6     <script
7     src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.16/angular.min.js"></script>
8 </head>
```

- **Step 2: Define ng-app**
  The second step you have to do in order to use the framework, is to define the ng-app you want to use. It is important to know, that the name has to match with the name which you assign to your module in your JavaScript-File.

```
1 <!DOCTYPE html>
2 <html ng-app="myApp">
3 <head>
```

- **Step 3: Create the view with HTML**

  After defining the ng-app you will use in your project, you can start with creating the view in the HTML-File. Afterwards you can add ng-directives of the AngularJS framework to enable Two-Way-Data-Binding.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="ISO-8859-1">
    <title>Example AngularJS</title>
    <script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.16/angular.min.js"></script>
</head>
<body>
    <div>
        <p>Name: <input type="text"></p>
        <h1>Welcome to our website!</h1>
    </div>
</body>
</html>
```

- **Step 4: Create your Script**

  Now you can finally start with implementing your controller. $scope carries data between all pieces of the system and thereby makes it possible to bind input fields to it. In this example only the username is stored and a function to welcome the new user. You can add as many variables as you want to your controller and use it in your model.

```
var app = angular.module("myApp", []);

app.controller("myController", function($scope){
    $scope.username = "";
    $scope.welcomeUser = function() {
        $scope.greeting = "Welcome " + $scope.username + "!";
    };
});
```

Here you can find a list of useful commands for your exercise:

$scope.myArray.push();	Add a new item to your array
$scope.myArray.length;	Get the length of your array
$scope.myArray.indexOf(item);	Get the index of an item of your array
$scope.items.splice(index, howmany);	Remove an item from your array

- **Step 5: Connect AngularJS with your View**

  The last thing that you have to do is to connect AngularJS with your view. In the last step you have created your controller to manage the data of your website. To get access to the data and to enable the Two-Way-Data-Binding you first need to embed your JavaScript-Controller in a script-tag. The *ng-app* directive already defines the AngularJS application. In order to define the controller you need to add the *ng-controller* directive to get access to the data.

  To read out data from your controller, you have to use expressions. Expressions are written inside double braces {{ }}. Inside the brackets you can write the name of the variable you want to show in your view.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="ISO-8859-1">
    <title>Example AngularJS</title>
    <script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.16/angular.min.js"></script>

</head>
<body>
    <div ng-controller="myController">
        <p>Name: <input type="text" ng-model="username"></p>
        <button type="submit" ng-click="welcomeUser()">Submit</button>
        <p>{{greeting}}</p>
    </div>

    <script src="AngularJSController.js"></script>
</body>
</html>
```